

TECH STYLE LTD

TECH-STYLE.CO

PRODUCT BROCHURE

pgrecon: Product Brochure

Free, open-source Oracle > PostgreSQL migration reconnaissance: 80 rules, effort estimates and offline analysis.

WHAT IS INSIDE

- The tables are the easy part
- Four moves: generate, extract, analyse, convert
- 80 rules across nine domains
- A number that shows its own arithmetic
- It converts what it can prove, and refuses the rest by name
- Six converters, one live PostgreSQL
- The whole surface area
- The document that gets the budget approved
- What it runs on
- Where this fits

Document	pgrecon brochure
Generated	16 October 2026
Online	https://www.tech-style.co/docs/pgrecon-brochure

Free, open-source migration reconnaissance. It answers the one question that decides whether an Oracle-to-PostgreSQL migration succeeds or overruns: what is actually in there, and what will it cost to move? It parses your PL/SQL with a real grammar, reports every incompatibility with the object and line where it was found, estimates the effort in person-days, and converts the schema it can prove into PostgreSQL DDL, all without ever connecting to your database.

```
$ pip install pgrecon
```

The tables are the easy part

Oracle licensing pushes organisations towards PostgreSQL, and the schema itself rarely stands in the way: tables, indexes and data move with well-trodden tooling. What sinks the timeline is twenty years of business logic: PL/SQL packages full of Oracle-only constructs that have no direct equivalent on the other side.

Migrations routinely run twelve to eighteen months and overrun their budgets for a single reason: nobody measured the code before committing to the date. The scary parts are discovered in month nine, by which point the plan is already written.

pgrecon is the measurement step. It reads the code you already have, tells you exactly which constructs will not survive the move, shows you where each one lives, and turns that into a defensible number. Run it before anyone signs anything.

We built it because we have done these migrations professionally, including for heavy-industry estates still running very old Oracle versions. The rules are the list of things that hurt us.

Status	v0.7.3 on PyPI · alpha
Licence	Apache 2.0, free for commercial use
Rules	80, each with fixture tests
Analysis	Fully offline, with no DB connection

It never touches your database.

pgrecon generates a SQL*Plus script of plain SELECTs that your own DBA reads and runs. You send back the output files; analysis happens on a laptop, against a local SQLite inventory. There is no listener to open, no account to create and nothing for a security team to sign off.

Parsed, not grepped.

The engine builds a real parse tree of your DDL and PL/SQL. It knows the difference between a dangerous construct in live code and the same word inside a comment or a string literal, which is where keyword-matching tools generate the noise that gets them ignored.

HOW IT WORKS

Four moves: generate, extract, analyse, convert

The only thing that runs near production is a read-only script your own DBA inspects first.

STEP 1**Generate the extraction script**

`pgrecon script --source-version 19` writes a SQL*Plus script tailored to your Oracle release. For 9.2 to 11.1 estates, `--legacy` produces a variant that runs on the older dictionary views.

STEP 2**Your DBA reads it, then runs it**

It is plain SELECT statements against the data dictionary, with nothing to approve in a change board meeting. `sqlplus readonly_user@service @pgrecon_extract.sql SCHEMA_NAME` produces a directory of dump files. SQL*Plus is the only requirement on the database server; no Python, no agent, no network access.

STEP 3**Load the dump into a local inventory**

`pgrecon load dump_dir --db inventory.db` parses the extract into a local database: every table, index, package, trigger, view, job and dependency, with the source text of each program unit.

STEP 4**Run the rules**

`pgrecon report --db inventory.db` walks the parse tree with all 80 rules and prints every finding: rule ID, severity from info to blocker, the object it lives in, and the evidence. Add `--remedies` for what to do about each one, or `--format json` to feed a tracker.

STEP 5**Price the work**

`pgrecon estimate --db inventory.db` converts the inventory and the findings into a low / expected / high range in person-days, and shows the arithmetic that produced it.

STEP 6**Convert what can be proved**

`pgrecon convert --db inventory.db` emits PostgreSQL DDL for everything it can carry faithfully, and a residue file naming every object it declined and why. Still offline, still from the same inventory.

STEP 7**Write the runbook for the data**

pgrecon runbook --db inventory.db does not move rows. It writes what the move needs: a data-only mover configuration, row-count and spot-sum validation SQL for both engines, the post-load sequence and materialised-view steps, and the cutover checklist.

STEP 8**Decide with evidence**

You now hold a defensible scope before a single line has been ported, and the same numbers can be re-run against a later extract to show progress.

WHAT IT FINDS**80 rules across nine domains**

Every rule ships with fixture tests, a severity, a remediation note and a documented explanation you can read with pgrecon explain RULE_ID.

pgrecon rule categories, counts and representative examples

DOMAIN		
	RULES	REPRESENTATIVE FINDINGS
PL/SQL code	18	Autonomous transactions, dynamic SQL, FORALL, collection types, the empty-string NULL trap
Schema objects	14	Database links and the remote calls made over them, scheduler jobs, materialised view logs, queues, evolved types, unparseable DDL
Storage	12	Interval partitioning, global temporary tables, IOTs, read-only tables, bitmap and function-based indexes
SQL constructs	13	CONNECT BY, Oracle outer-join syntax, ROWNUM, MERGE, SYS_CONTEXT, the MODEL clause, PIVOT, flashback queries
Data types	9	LONG, XMLTYPE, ROWID, BFILE, SDO_GEOMETRY, BYTE-semantics strings, TIMESTAMP WITH LOCAL TIME ZONE
System packages	5	UTL_FILE, UTL_HTTP / SMTP / TCP, DBMS_SQL, DBMS_LOB, DBMS_OUTPUT
Performance	5	Optimiser hints, global indexes on partitioned tables, plan baselines, query-rewrite MVs
Environment	2	Character-set encoding decision, object grants to migrate

DOMAIN

RULE
S

REPRESENTATIVE FINDINGS

Packages 2 Package-level state, initialisation blocks

Evidence, not adjectives

Each finding names the object and the location inside it, so an engineer can open the code and see the problem rather than take the tool's word for it.

Severity that means something

Findings run from info to blocker. A blocker is something with no PostgreSQL equivalent that will require a design decision, beyond a mere syntax difference.

A remedy per rule

--remedies prints what to do about each finding: the PostgreSQL equivalent, the extension that covers it, or the redesign it forces.

Parse failures are findings

If the parser cannot read a program unit, that is reported as a finding and never silently skipped. Unknown code is a risk, and it is accounted for as one.

Deterministic by design

Same dump in, same findings out. Two people running the same extract get the same report, which is what makes it usable in a commercial conversation.

JSON for the rest of your stack

--format json exports the full finding set for a backlog, a spreadsheet or a dashboard, so the assessment survives the meeting it was made for.

Measured at estate scale

A synthetic estate of 5,000 tables and 100,000 lines of PL/SQL across 1,600 stored units (one of them a 16,000-line package body) loads and deep-parses in under 40 seconds on a laptop, and reports in about two. The generator ships with the source, so the measurement is reproducible rather than quoted.

Fuzzed, not merely tested

An adversarial dump generator throws names past 63 bytes in ASCII and Hangul, names that collide once Oracle's namespaces fold into PostgreSQL's, reserved words as columns, every partition layout and spools in a foreign code page. CI runs a fresh window of seeds nightly and checks that every object the loader stored is either created in the DDL or named in the residue.

A number that shows its own arithmetic

An estimate nobody can interrogate is worth nothing in a budget meeting. pgrecon estimate builds its range from five components (baseline and environment setup, schema conversion, remediating the findings it just reported, porting PL/SQL by volume, and moving the data) and prints the contribution of each.

The output is a low, expected and high figure in person-days, with the person-month equivalent. Because the workings are visible, your team can argue with any line of it and re-run with their own assumptions instead of discarding the whole thing.

Calibration is the honest caveat: the model is built from real project experience, but it does not know your team, your test estate or your change-control overhead. That judgement is what our assessment report adds.

```
# estimate from the bundled sample inventory pgrecon estimate --db sample.db Migration
effort estimate (person-days) baseline and environment 5.0 schema conversion 1.7 finding
remediation 74.3 PL/SQL porting by volume 0.8 data movement 0.0 development subtotal
81.7 With testing and stabilization: low 106, expected 131, high 180 person-days (5.1 to
8.6 person-months)
```

It converts what it can prove, and refuses the rest by name

Since v0.2 the same inventory drives a converter, and it now covers schema structure, views, materialised views, triggers, grants and comments end to end on the test estates. pgrecon convert writes two files, offline like everything else. The first is PostgreSQL DDL for what it can carry faithfully: tables under a documented type mapping (NUMBER stays exact, never a float), keys, checks, foreign keys, secondary indexes, native partition children for range, list, hash and composite layouts, views transpiled with (+) joins folded to ANSI, sequences restarted at their extracted position, synonyms as views, database links scaffolded as oracle_fdw servers, generated columns, and standalone functions and procedures whose every construct has a provably equivalent PL/pgSQL form, with comments and formatting carried through and SELECT INTO made STRICT so NO_DATA_FOUND still raises.

Inside those routines the top-N idiom converts (a ROWNUM bound over a sorted subquery becomes LIMIT), and MERGE carries over with its action conditions moved onto the WHEN clauses, the SET aliases PostgreSQL rejects dropped, and USING dual made a one-row source. Oracle's date-plus-number arithmetic becomes an explicit INTERVAL, and views that sort, group or compare XML or JSON columns, which PostgreSQL cannot order the way Oracle does, are declined by name.

The second file is the residue: one line per declined object, naming the construct and the line number. Packages, CONNECT BY, autonomous transactions, REF CURSOR interfaces, BULK COLLECT, a ROWNUM sitting beside an ORDER BY: the work that needs a person is refused by name rather than guessed at, and a routine that calls a refused routine is refused with it.

```
# convert from the same inventory the report came from pgrecon convert --db inventory.db
schema_pg.sql what converts, provably schema_residue.txt what does not, named and located
```

Nothing invalid ships, and nothing is lost silently.

CI applies the bundled sample's conversion to a live PostgreSQL 16, 17 and 18 on every commit, with check_function_bodies on. Whatever the converter cannot carry faithfully becomes a named residue line instead of quietly wrong output, so the gap between "converted" and "done" is a file you can count before testing starts.

BENCHMARK

Six converters, one live PostgreSQL

Every converter claims to convert an Oracle schema. The benchmark asks a narrower question: apply the tool's own output to a real PostgreSQL 16, statement by statement, and count how many statements PostgreSQL itself rejects. A missing feature or a style complaint does not count; a statement the target database refuses does. That is the bar a real migration hits in production, whether the tool warned about it or not.

Statements rejected by PostgreSQL 16 when each converter's output was applied to the benchmark's Oracle schemas, August 2026, with pgrecon re-measured on 2026-10-05

CONVERTER	STATEMENTS REJECTED	NOTES
pgrecon 0.7.3	0	Of 423 statements on the eight public schemas, re-measured in CI on 2026-10-05, and 0 in the August nine-schema round. Every decline is a named residue line
CYBERTEC ora_migrator	40	Plus two whole-schema crashes; PL/SQL is out of its scope by design and is not counted against it
Ora2Pg v25	77	Clean on the simple estates; needed two accommodations to apply at all
EDB Migration Toolkit 55.13	135	Of 477 emitted statements, with entire package libraries omitted without a word and reported as success
AWS SCT	481	Clean on HR only; surviving objects depend on the aws_oracle_ext runtime extension

CONVERTER

STAT
EME
NTS
REJE
CTED

NOTES

Dalibo PostgreSQL Migrator 1.0	58	Of 445 statements on the eight public schemas, run once on 2026-09-19; emits no views or code
---	----	---

Zero rejections is not the same as everything converts.

At 0.7.3 pgrecon creates 241 of the 594 answerable objects in the eight public schemas (41%) and declines the other 353 by name, losing none. That is above 95% on the business-shaped HR and CO samples, 81% on Logger and far below on the object-type and package showcases, which nothing converts mechanically. (The August figure of 51% counted every object across all nine schemas, lab schemas included.) The difference is that every unconverted object is one line in the residue report with a reason, so the gap is a work list you can price rather than a surprise in testing.

COMMANDS

The whole surface area

One command per step, and nothing hidden behind a service.

pgrecon command reference

pgrecon script	Generate the SQL*Plus extraction script. --source-version targets your Oracle release; --legacy produces the 9.2 to 11.1 variant.
pgrecon load	Parse a dump directory into a local inventory database. --encoding handles dumps that are not UTF-8.
pgrecon report	Run all rules and print the findings. --remedies adds remediation guidance; --format json exports the full set.
pgrecon explain	Print the full documentation for a single rule ID: what it detects, why it matters and how to resolve it. With no argument it lists the whole catalogue with severities.
pgrecon estimate	Produce the low / expected / high effort range in person-days, with the component breakdown.
pgrecon convert	Emit PostgreSQL DDL for everything provably convertible, plus a residue file naming every object it declined and where it lives.
pgrecon runbook	Write the data-movement artifacts, offline: a data-only mover configuration, row-count and spot-sum validation SQL for both engines, the post-load sequence and materialised-view steps, and the cutover checklist.

pgrecon info

Summarise an inventory: its metadata, a count of every object type it holds, and how much of the DDL parsed.

```
pip install pgrecon # no Oracle yet? the repository carries a sample dump git clone
https://github.com/Muzzammil242/pgrecon pgrecon load pgrecon/examples/dump_oracle21c --
db sample.db # or run it against your own estate pgrecon script --source-version 19
# ... your DBA runs the generated script and returns dump_dir/ pgrecon load dump_dir --
db inventory.db pgrecon report --db inventory.db --remedies pgrecon estimate --db
inventory.db pgrecon convert --db inventory.db pgrecon runbook --db inventory.db
```

MIGRATION ASSESSMENT REPORT

The document that gets the budget approved

The free tool answers the engineer's question: what is in there and what breaks. The assessment report answers the board's: what does this cost, how long does it take, in what order, and what could go wrong.

What you receive

- An executive summary a non-technical sponsor can act on
- Every finding prioritised and rewritten in plain English, with business impact
- A phased migration roadmap: what moves first, what moves last, and why
- A testing and cutover plan, including how you prove equivalence
- A cost and effort estimate calibrated against real migration projects, not just the tool's model
- The risk register: what we would worry about in your specific estate
- A walkthrough call with the engineer who wrote it

How an assessment runs

Typical turnaround is two weeks from receiving the dump. The fee is credited against a subsequent migration engagement if you choose to go ahead with us.

Nobody is locked in.

The detection engine is Apache 2.0 and stays that way. You can run it forever, fork it, or hand it to another supplier. What we sell is the judgement layered on top and the hands that do the migration.

STACK

What it runs on

pgrecon technical specification

Runtime	Python 3.11 or later, installed from PyPI with <code>pip install pgrecon</code> .
Source Oracle	11.2 and later with the standard script, verified nightly in CI against Oracle XE 21c and Oracle Free 23ai. Oracle 9.2 to 11.1 via <code>pgrecon script --legacy</code> , itself verified nightly in CI against Oracle XE 11g.
Target	PostgreSQL. Rules are written against stock PostgreSQL, noting where an extension covers the gap.
On the database server	SQL*Plus only. No Python, no agent, no outbound connection.
Analysis store	A local single-file inventory database built from the dump.
Analysis method	Grammar-based parsing of DDL and PL/SQL into a parse tree, with no keyword matching.
Output	Terminal report with severities and remedies, JSON export, per-rule explanations, effort estimate, converted PostgreSQL DDL with a named residue file, and a migration runbook.
Testing	Every rule carries fixture tests; the suite runs against real Oracle extracts, the converter's output is applied to live PostgreSQL 16, 17 and 18 on every commit, and a fresh window of fuzz seeds runs nightly.
Licence	Apache License 2.0. Issues and pull requests welcome.

RELATED

Where this fits

SERVICES

Oracle > PostgreSQL migration

Assessment, schema and PL/SQL conversion, data movement, cutover and support.

SERVICES

Data engineering & BI

What happens to the reporting layer once the database has moved.

OPEN SOURCE

Our other packages

Everything else we publish on PyPI, npm and GitHub.

CONTACT**Talk to us**

Tech Style Ltd is a UK technology company building enterprise software, procurement intelligence and AI platforms. Tell us what you are trying to do and we will tell you honestly whether we are the right people for it.

This document online	www.tech-style.co/product-pgrecon
Website	www.tech-style.co
Enquiries	hello@tech-style.co
Products	www.tech-style.co/products
Partner programme	www.tech-style.co/partners

This document was generated on request.

It is built from the live content of tech-style.co at the moment you downloaded it, so it can never quote a price, a version or a status the site has since moved on from. Request a fresh copy any time at www.tech-style.co/downloads.